Enoncés des travaux dirigés de VBA, première année

Table des matières

1 TD 1 : Equation différentielle stochastique et graphique

(correction page 29)

Quelques remarques en préambule, il ne faut pas hésiter à se promener dans l'aide de Microsoft concernant VBA, en appuyant sur la touche F1 par exemple. Internet est également une bonne source, il suffit parfois de recopier un message d'erreur obscur dans Google pour obtenir une réponse. Une autre méthode simple consiste à enregistrer sous forme de macro l'action que vous voulez programmer. Microsoft convertit toute marco sous la forme d'un code écrit en VBA dont il est facile de s'inspirer.

VBA n'est pas un langage compliqué, la plupart du temps est passée à chercher comment faire une action précise comme ajouter un graphe automatiquement dans une feuille Excel. Les quatre TD n'ont pas pour but de vous apprendre à programmer mais plutôt de vous montrer quelques usages courant des logiciels Microsoft.

—

Objectifs de ce TD :

1. Découvrir VBA
2. Résoudre une équation différentielle stochastique avec VBA
3. Tracer un graphique

—

L'équation différentielle stochastique de Black et Scholes apparaît souvent lorsqu'on parle de finance :

$$\frac{dY_t}{Y_t} = rdt + \sigma dW_t \tag{1.1}$$

Oô :

Y_t	est le prix à l'instant t d'une action
r	est le taux d'intérêt sans risque
σ	est la volatilité du prix de l'action
t	représente le temps, dt un petit intervalle de temps
W_t	est un mouvement browien, et plus simplement dans ce TD une variable aléatoire normale de moyenne nulle et de variance t, dW_t est une variable aléatoire normale de moyenne nulle et de variance dt.

Cette équation différentielle, si on la résoud sans tenir compte de sa partie stochastique donne $Y_t = Y_0 e^{rt}$. Avec un taux d'intérêt positif, le prix d'une action augmente. Mais cela n'est jamais beau et on est obligé de tenir compte des aléas plus ou moins importants selon la valeur de σ. Plus on avance dans le temps, plus la part du hasard est importante et plus la probabilité que le prix de l'action suive la courbe $Y_t = Y_0 e^{rt}$ est faible.

Si on cherche à résoudre l'équation différentielle stochastique de manière numérique entre les temps 0 et T, on procède comme pour une équation différentielle classique. On considère donc que dt est un petit intervalle de temps et on écrit l'équation comme ceci :

$$Y_{t+dt} = Y_t \left(1 + rdt + \sigma dW_t\right) \tag{1.2}$$

Ensuite, à partir de Y_0 qui est connu, on calcule les valeurs Y_{dt}, Y_{2dt}, Y_{3dt}, ..., Y_{T-dt}, Y_T. La différence avec une équation différentielle non stochastique est qu'il n'y a plus une seule solution mais une infinité : à chaque itération pour calculer, $Y_{(n+1)dt}$ en fonction de Y_{ndt}, on utilise un dW_t différent pris au hasard.

Ceci signifie que Y_T n'est pas une constante mais une variable aléatoire. Il faut donc calculer un grand nombre de solutions pour obtenir sa moyenne, sa variance, son maximum, son minimum... Et c'est l'objectif de cette séance.

1) Pour commencer, il faut lancer l'application Excel accessible depuis le menu Démarrer (ou Start) de Windows. Ensuite, il faut recopier exactement ce qu'il y a dans la figure 1.1. Il est préférable de ne pas se tromper de ligne ou de colonne pour que la suite de l'énoncé soit cohérente avec votre travail. N'oubliez pas de sauver le fichier avec un nom de votre choix. Par la suite, sauvez régulièrement au cas où la mauvaise humeur d'Excel l'amènerait à *planter* en pleine programmation ce qui ne devrait normalement pas vous arriver durant de TD, fort heureusement.

	A	B	C	D	E
1	**Black Scholes**				
2					
3	r	sigma	x0	dt	T
4	1	1	1	0,1	10

FIG. 1.1 – Début pour la question 1.

2) La suite va se passer dans une autre fenêtre : celle où l'on écrit les programme en VBA. La figure 1.2 montre comment ouvrir l'éditeur VBA qui est aussi accessible grâce au raccourci ALT+F11. Il faut ensuite ouvrir une fenêtre ou *module* permettant de saisir le programme VBA (seconde image de la figure 1.2).

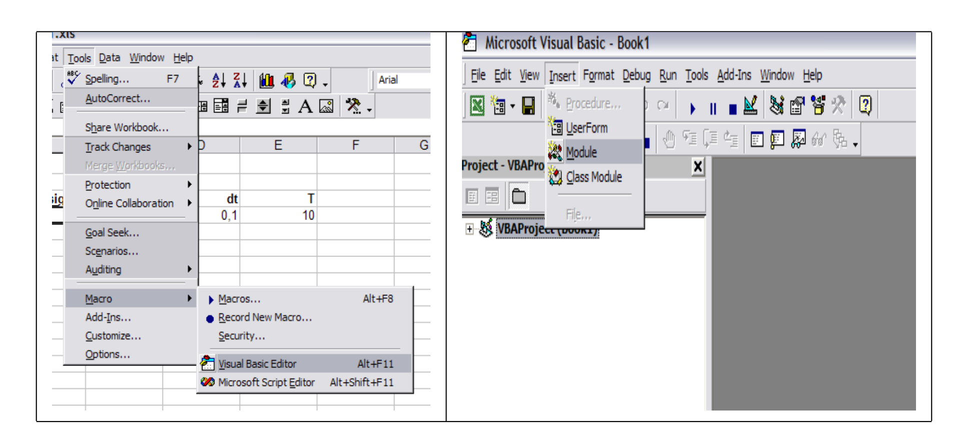


FIG. 1.2 – Création d'un module : fenêtre où va être écrit le programme VBA.

Une nouvelle fenêtre apparaît. Il faut maintenant saisir le programme de la figure 1.3.

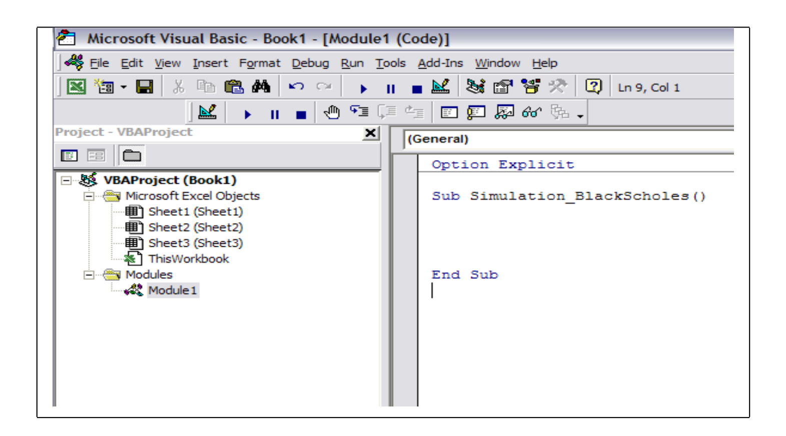


Fig. 1.3 – Prémisses d'un programme.

L'instruction `Option Explicit` indique au langage VBA qu'il doit être plus stricte lors de sa compréhension du programme, cela évite qu'il interprète mal une instruction, il détectera une erreur à la place.

Une fois que ce petit programme est tapé, on écrit à l'intérieur de la procédure (entre les instructions `Sub` et `End Sub` la ligne :

```
MsgBox "boîte à message"
```

Placez ensuite le curseur sur cette ligne et cliquez sur l'icône en forme de triange orienté vers la droite dans la barre d'outils. Une boîte à message doit normalement apparaître à l'écran : le programme est en train de tourner et vous demande de presser le bouton *Ok* pour conclure.

3) On cherche maintenant à récupérer les valeurs présentes dans la feuille Excel de la question 1. Il y a 5 informations, on doit donc créer 5 variables. On considère que ce sont toutes des variables réelles et pour en déclarer une, on écrit[1] :

```
Dim r As Double
```

On appelle les 5 variables à récupérer `r`, `sigma`, `x0`, `dt`, `T`. Une fois que celle-ci sont créées, on peut leur affecter une valeur en récupérant celles présentes dans la feuille Excel[2] Pour récupérer le contenu d'une case, on utilise :

```
r = Worksheets("Sheet1").Cells(4, 1).Value
```

N'oubliez de déclarer vos variables avant de leur affecter une valeur. Il en sera de même plus tard pour les variables utilisées dans les boucles `For`.

`Sheet1` est le nom de la feuille. Si la version d'Excel que vous utilisez est française, ce sera un autre nom comme `Feuil1`. L'instruction précédente récupère donc la valeur de la case ligne 4, colonne 1 de la feuille `Sheet1`. On procède de même pour `sigma`, `x0`, `dt`, `T`.

4) On va maintenant créer une fonction permettant de calculer une solution de l'équation différentielle stochastique de Black et Scholes. Elle prend comme paramètres :

1. `r` : un réel ou `Double` en VBA

[1] Le nom des variables, comme celui des fonctions ne peut inclure ni espace, ni accent. Aucune distinction n'est faite entre minuscules et majuscules. Il n'est pas besoin d'être très rigoureux sur les minuscules et majuscules, VBA les corrige lorsqu'elles ne correspondent pas au nom utilisé lors de la déclaration de la variable.

[2] Avec l'instruction `Option Explicit`, il est impossible d'utiliser une variable sans que celle-ci ait été préalablement déclarée.

2. `sigma` : un réel ou `Double` en VBA
3. `x0` : un réel ou `Double` en VBA
4. `dt` : un réel ou `Double` en VBA
5. `T` : un entier ou `Long` en VBA

Elle retournera un tableau de valeurs correspondant au vecteur $(Y_0, Y_{dt}, Y_{2dt}, ..., Y_T)$. Le type de résultat en VBA est un `Variant`. Pour déclarer la fonction, on utilise la syntaxe suivante :

```
Function Simulation(ByVal r As Double, ByVal sigma As Double, ByVal x0 As Double, _
            ByVal dt As Double, ByVal T As Long) As Variant

    '  code de la fonction

    ' Lorsqu'on connaît le résultat de la fonction, on écrit
    Simulation = résultat

End Function
```

Le blanc souligné _ à la fin de la première ligne permet d'écrire une instruction sur deux lignes et non sur une seule comme le langage VBA l'impose. Le blanc souligné doit toujours être précédé d'un **espace**. Le mot-clé `ByVal` signifie qu'on passe les paramètres par valeur : ils sont recopiés. L'inverse est `ByRef` qui signifie un passage par adresse : ils ne sont pas recopiés et peuvent être modifiés dans la fonction. Le type du résultat apparaît en fin de ligne. L'apostrophe sert à insérer un commentaire dans le programme. Cette fonction doit être insérée juste après `Option Explicit` et avant la procédure qui suit et qui devra faire appel à cette fonction.

5) On cherche à calculer une solution de l'équation de Black et Scholes. Cette solution est un tableau qui contient $n = \frac{T}{dt} + 1$ valeurs.

```
    Dim n as Long
    n = T / dt + 1
```

On déclare un tableau de réels dont les indices vont de 0 à `n`. La première valeur est `x0`.

```
    Dim res() As Double
    ReDim res(n)
    res (0) = x0
```

Pour simuler une variable normale de moyenne nulle et de variance `dt`, on utilise le code suivant :

```
    Dim w As Double
    w = Rnd     ' nombre aléatoire de loi uniforme [0,1]
    w = Application.WorksheetFunction.NormSInv(w) * dt ^ 0.5
```

La fonction `NormSInv` est une fonction d'Excel, il suffit d'aller voir l'aide pour savoir ce qu'elle fait. D'une manière générale, pour utiliser une fonction dans un programme VBA alors qu'on a l'habitude de se servir dans une feuille de calcul, il faut utiliser le préfixe : `Application.WorksheetFunction`.

Il ne reste plus qu'à savoir comment faire une boucle pour terminer la fonction :

```
    For i = 1 To nb
       res (i) = res(i-1) * ... à compléter
    Next i
```

Enfin, lorsque le tableau est complet et qu'il est le résultat de la fonction `Simulation`, il ne reste plus qu'à ajouter à la fin la ligne :

```
Simulation = res
```

Il ne reste plus qu'à écrire le code complet de cette fonction. Un dernier indice toutefois, intéressez-vous particulièrement à l'emplacement des deux lignes qui génère un nombre aléatoire.

6) On désire maintenant recopier les valeurs d'une solution dans la feuille Excel de départ. La première étape consiste à récupérer le tableau calculé par la fonction de la question précédente :

```
Dim solution As Variant
solution = Simulation(r, sigma, x0, dt, T)
```

On récupère le nombre d'éléments dans `solution` :

```
Dim nb as Long
nb = UBound (solution)
```

On recopie les valeurs dans la feuille, celles-ci doivent être réparties sur deux colonnes : le temps t et Y_t.

```
For i = 0 To nb
    Worksheets("Sheet1").Cells(7 + i, 1) = ... ' temps t
    Worksheets("Sheet1").Cells(7 + i, 2) = ... ' Y_t
Next i
```

Une fois que tout cela est fait, on place le curseur dans la procédure principale et on clique sur le même triangle que tout-à-l'heure. Si tout se passe bien, la résultat apparaît sur la feuille Excel. Sinon, c'est le moment d'appeler le chargé de TD pour corriger les erreurs et pour jouer avec le *debugger* qui permet d'afficher le contenu des variables en cours d'exécution.

7) C'est le petit moment de détente avant de retourner à la programmation : on veut associer un bouton de la barre d'outils à la procédure (ou macro) qu'on vient de créer. Un clic droit dans la barre d'outils fait apparaître l'image de la figure 1.4. On clic sur *Customize*. La seconde image de la figure 1.4 apparaît. Il faut s'arranger pour faire apparaître *Custom Button*. En maintenant le bouton gauche appuyé, on fait glisser le bouton jusqu'à la barre d'outils où il s'insère. Une fois posé, on presse le bouton droit dessus et on voir apparaître le troisième image de la figure 1.4.

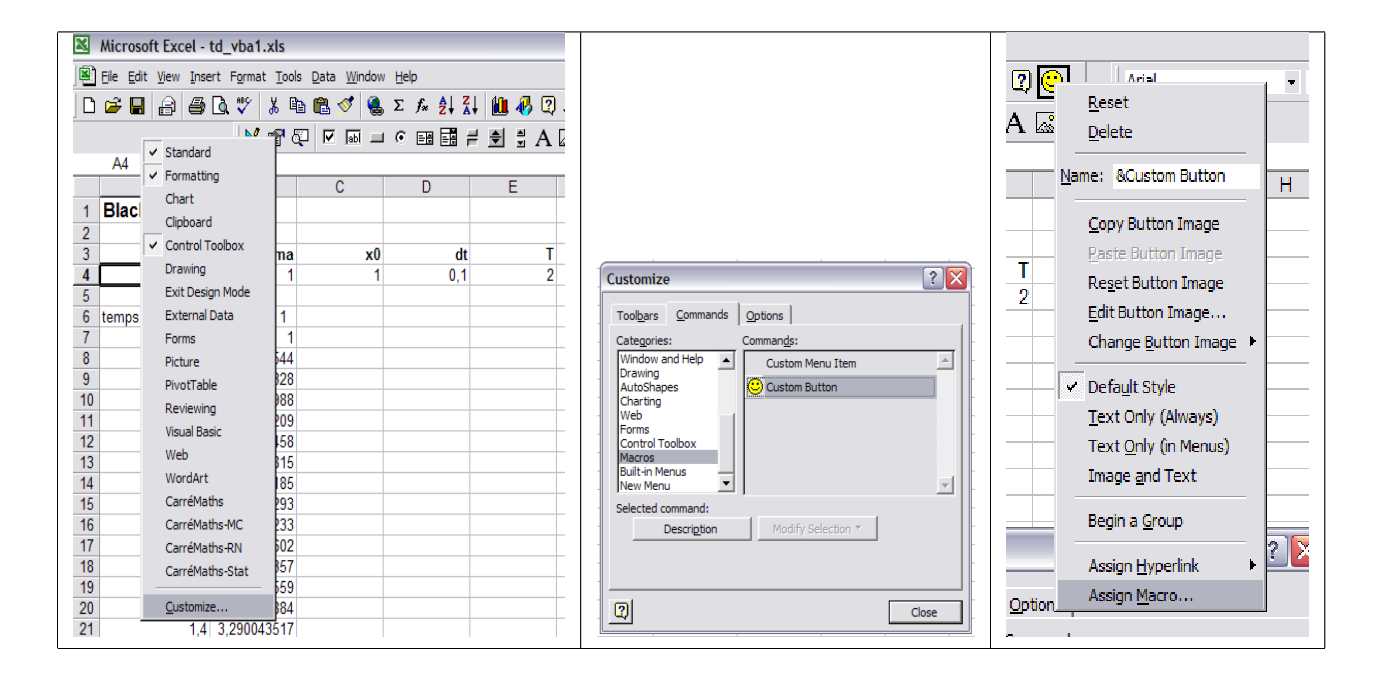


Fig. 1.4 – Barre d'outils et macro.

La fenêtre de la figure 1.5 apparaît. Il ne reste plus qu'à sélectionner votre macro et à cliquer sur *Ok*. On ferme ensuite la première fenêtre. Et, enfin, on clique sur le bouton ajouté : normalement, le contenu de la feuille Excel change : une nouvelle solution apparaît. Les valeurs de la solution précédente ont été remplacées par une autre, un second graphique apparaît.

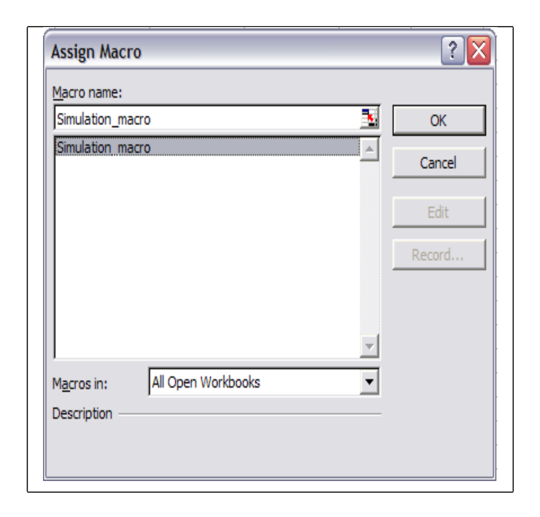


Fig. 1.5 – Un nouveau bouton.

8) La dernière épreuve consiste à créer un graphe à partir de la solution précédemment calculée. On récupère d'abord la plage contenant les données pour le graphe.

```
Dim plage As Range
Set plage = Worksheets("Sheet1").Range(_
                   Worksheets("Sheet1").Cells(7, 1), _
                   Worksheets("Sheet1").Cells(8, 2) _
               )
```

L'instruction *Set plage* = ... permet d'associer à `plage` une partie de la feuille Excel sans la recopier. Dans ce cas, sans le mot-clé `Set`, l'affectation est impossible. En fait, le type `Range` désigne une partie d'une feuille Excel, toute variable de ce type doit faire référence à un ensemble de vraies cases, c'est pourquoi on ne peut pas les recopier. Il en va de même pour de nombreux objets d'Excel tels que les graphes. L'instruction `Set` permet en faire de donner un nom plus court à quelque chose de long.

On créé ensuite un graphe dans la feuille `Sheet1` :

```
Dim graphe_in As ChartObject
Dim graphe As Chart
Set graphe_in = Worksheets("Sheet1").ChartObjects.Add(100, 30, 400, 250)
Set graphe = graphe_in.Chart
```

$100, 30, 400, 250$ sont les coordonnées du graphe dans la feuille. Le graphe est désigné par la variable `graphe`. On spécifie maintenant son type : XY.

```
graphe.ChartType = xlXYScatterLines
```

On précise quelles sont les données et qu'elles sont présentées en colonnes :

```
graphe.SetSourceData plage, xlColumns
```

On précise qu'il a un titre et quel est-il.

```
graphe.HasTitle = True
graphe.ChartTitle.Text = "Black Scholes"
```

On précise qu'il a des axes et la légende de ces axes.

```
graphe.Axes(xlValue, xlPrimary).HasTitle = True
graphe.Axes(xlValue, xlPrimary).AxisTitle.Text = "Xt"

graphe.Axes(xlCategory, xlPrimary).HasTitle = True
graphe.Axes(xlCategory, xlPrimary).AxisTitle.Text = "temps (jour)"
```

On modifie le titre de la série et son apparence :

```
graphe.SeriesCollection(1).Name = "Courbe1"
graphe.SeriesCollection(1).Border.Color = RGB(0, 0, 255)
graphe.SeriesCollection(1).MarkerStyle = xlMarkerStyleNone
```

Et c'est fini ou presque pour les plus courageux qui peuvent essayer de tracer plusieurs solutions dans un même graphe et pour y arriver, voici quelques lignes qui permettent d'ajouter une série à un graphe.

```
Dim serie
Set serie = graphe.SeriesCollection.NewSeries
serie.XValues = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1), _
                    Worksheets("Sheet1").Cells(7 + nb, 1))
serie.Values = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1 + i), _
                    Worksheets("Sheet1").Cells(7 + nb, 1 + i))
serie.Name = "nouvelle série"
```

Lorsqu'on ne connaît le type de la variable à déclarer comme pour `serie`, `Dim serie` suffit sans indication de type. Néanmoins, pour des types simples (`Long`, `String`, `Double`, il est préférable de préciser ce type, le programme est plus rapide.

2 TD 2 : Construction d'une fichier Acrobat et envoi par email

(correction page 33)

Un professeur de maths a instauré une coutume lors de la dernière séance de la semaine. Le vendredi est consacré à la correction d'un examen ou d'exercices. Ce professeur a décidé de regrouper les élèves par deux : le premier a obtenu une bonne à l'examen précédent tandis que le second a obtenu une mauvaise note. Il a expliqué aux élèves qu'il leur enverrait par email tous les mercredi la composition des groupes de la séance du vendredi suivant afin qu'ils ne perdent pas trop de temps à s'installer dans la salle.

Au bout de quelques semaines, cette tâche est devenue assez contraignante, elle nécessitait vingt minutes à chaque fois, vingt-cinq minutes s'il fallait rallumer l'ordinateur et se reconnecter à internet parce qu'il avait oublié. Ce professeur décida donc d'optimiser la tâche à partir d'une feuille Excel regroupant les notes des élèves et leur adresse email. Il a isolé les grandes étapes de son modeste programme :

1. tri des notes par ordre croissant
2. composition des groupes, le premier avec le dernier, le second avec l'avant-dernier et ainsi de suite
3. création d'un fichier Word contenant le résultat
4. conversion de ce fichier en un fichier Acrobat (PDF) que les élèves peuvent imprimer mais pas modifier
5. envoi de ce fichier à tous les élèves

Malgré l'ampleur de la tâche, ce professeur s'est dit qu'en perdant quelques heures pour concevoir le programme, il gagnerait vingt minutes chaque semaine, soit une quinzaine d'heures sur l'année, sans compter les heures qu'il gagnerait l'année prochaine si l'expérience s'avère fructueuse. Presque surpris, c'est plein d'entrain qu'il s'est mis à programmer.

1) La première étape consiste à construire une feuille Excel avec quelques élèves, leurs notes et leur email. On pourra s'inspirer de la figure 2.6. Il est conseillé de choisir un nombre pair d'élèves. Si vous utilisez de vrais emails, il est probable que vous ayez à nettoyer un peu votre boîte de message après cette séance de TD, voire beaucoup selon l'humour.

	A	B	C
1	**Prénom**	**Note**	**Mail**
2	Archibald	12	Archibald@math.fr
3	Aladdin	4	Aladdin@math.fr
4	Albert	18	Albert@math.fr
5	Alphonse	11	Alphonse@math.fr
6	Candy	7	Candy@math.fr
7	Cendrillon	1	Cendrillon@math.fr
8	Constance	2	Constance@math.fr
9	Cunégonde	6	Constance@math.fr
10	Gertrude	7	Gertrude@math.fr
11	Gontrand	6	Gontrand@math.fr
12	Hubert	8	Hubert@math.fr
13	Téophile	15	Téophile@math.fr
14			

Fig. 2.6 – Début pour la question 1.

2) La première tâche consiste à trier les données ou de préfèrence une copie des données au cas où le programme ne serait pas bon du premier coup. On découpe cette épreuve en quatre :

1. sélection des informations à copier
2. copie
3. sélection de la copie
4. tri

Plutôt que de vous donner la syntaxe de chacune des commandes, l'objectif est d'enregistrer une macro faisant exactement ces quatre tâches et de se servir du code produit par Excel comme début de programme.

3) On s'occupe de la seconde tâche : composer les groupes. Cette partie ne nécessite pas de connaissance particulière d'Excel à part lire ou écrire le contenu d'une case. Il suffit d'utiliser pour cela la fonction `Cells` dont l'aide de Microsoft vous fournira le mode d'emploi.

4) Il faut maintenant construire un fichier Word. C'est la partie la plus longue dont voici les détails :

1. On sélectionne le tableau des couples. On s'inspire de la question 2.
2. On le copie. On s'inspire aussi de la question 2.
3. On le colle dans un nouveau document Word.

```
' on crée un document Word
Dim word As Object
Set word = CreateObject("Word.Application")
word.Documents.Add

' on met en forme
word.Selection.Font.Size = 16
word.Selection.Font.Bold = True

' écriture d'un petit texte dans ce nouveau document et on va deux fois à la ligne
word.Selection.TypeText "Organisation de la classe vendredi prochain" _
            & vbCrLf & vbCrLf

' vbCrLf insère un paragraphe dans le document Word
' le symbole & permet de concaténer plusieurs chaînes de caractères

' on ajoute le tableau dans le document word
word.Selection.Paste

' on change la mise en forme
word.Selection.Font.Size = 12
word.Selection.Font.Bold = False

' on écrit un petit message de fin
word.Selection.TypeText vbCrLf & vbCrLf & "Bon courage et à vendredi" _
   & vbCrLf & vbCrLf & "Votre professeur de mathématiques"
```

4. On imprime de document dans un fichier. Il faut vérifier que l'imprimante *HP LaserJet 5P/5MP PostScript* est disponible sur votre ordinateur (aller dans le menu imprimer). Sinon, il faut choisir une imprimante PostScript comme *Garamond**Klee* à l'ENSAE.

```
' on imprime le document dans un fichier posscript
' l'imprimante doit être installée sur votre ordinateur
word.ActivePrinter = "HP LaserJet 5P/5MP PostScript"
```

```
    word.PrintOut Filename:="w:\vendredi.doc", PrintToFile:=True, _
               OutputFileName:="w:\vendredi.ps"

    ' Fermeture de ce document :
    word.ActiveDocument.Close
    Set word = Nothing
```

5) Il faut convertir le fichier vendredi.ps en vendredi.pdf parce que Microsoft ne sait pas produire directement un fichier Acrobat[3]. Deux options sont possibles, la première utilise la version Windows de latex :

```
Shell ("C:\texmf\miktex\bin\epstopdf.exe w:\vendredi.ps")
```

La seconde solution, parfaite pour l'ENSAE, utilise le logiciel gratuit GhostScript :

```
Dim s As String
s = Chr(34)
Shell (s & "C:\Program Files\Ghostgum\gs8.51\bin\gswin32c" & s & _
    " -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dPDFSETTINGS=/prepress -sOutputFile=" & _
    "w:\vendredi.pdf d:\vendredi.ps")
```

6) Il ne reste plus qu'à envoyer le mail à tous les élèves. Quelques indices...

```
Dim look As Object
Set look = CreateObject("Outlook.Application")
Dim email As Object
Set email = look.CreateItem(olMailItem)
email.to = ...
email.Body = "organisation de la séance de vendredi, voir pièce jointe"
email.Subject = "cours de maths"
email.Attachments.Add "w:\vendredi.pdf"
email.Send
```

Il ne reste plus qu'à déterminer l'adresse à laquelle sera envoyée le mail, c'est-à-dire la concaténation de toutes les adresses des élèves séparées par un point virgule.

7) Il est possible comme en Python de prendre en compte le fait que le programme qu'on vient d'écrire ne marche pas. Par exemple, il marche sur un ordinateur mais pas sur un autre parce que les logiciels nécessaires ne sont pas tous installés. Dans ce cas, VBA propose la syntaxe suivante :

```
Sub procedure_protege_contre_les_erreurs ()

    On Error Goto Marqueur1

       ' s'il se produit une erreur dans ces lignes,
       ' le programme va directement à la ligne qui
       ' comment par Marqueur1

    On Error Goto Marqueur2
```

[3] Open Office, l'équivalent gratuit de Microsoft Office, le fait très bien. Il est probable que les prochaines versions de Microsoft Office le fassent également.

```
        ' s'il se produit une erreur dans ces lignes,
        ' le programme va directement à la ligne qui
        ' comment par Marqueur2

        ' la procédure s'arrête ici
        Exit Sub

Marqueur1 :

    ' on affiche un message d'erreur
    MsgBox "Une erreur s'est produite dans la première partie."
    Exit Sub

Marqueur2 :

    ' on affiche un message d'erreur
    MsgBox "Une erreur s'est produite dans la seconde partie."
    Exit Sub

End Sub
```

Il est fort probable que le programme écrit à l'ENSAE ne marche pas ailleurs parce que l'imprimante utilisée n'est pas disponible, que le programme de conversion au format Acrobat n'existe pas ou n'est pas au même endroit. Il est préférable d'afficher un message explicite dans ces cas-là plutôt que le programme ne s'arrête et laisse apparaître l'éditeur VBA.

8) Pour terminer, on se propose d'ajouter un bouton dans la feuille Excel qui permet d'exécuter le programme sans passer par l'éditeur VBA. Suivez les instructions des figure 2.7 et 2.8.

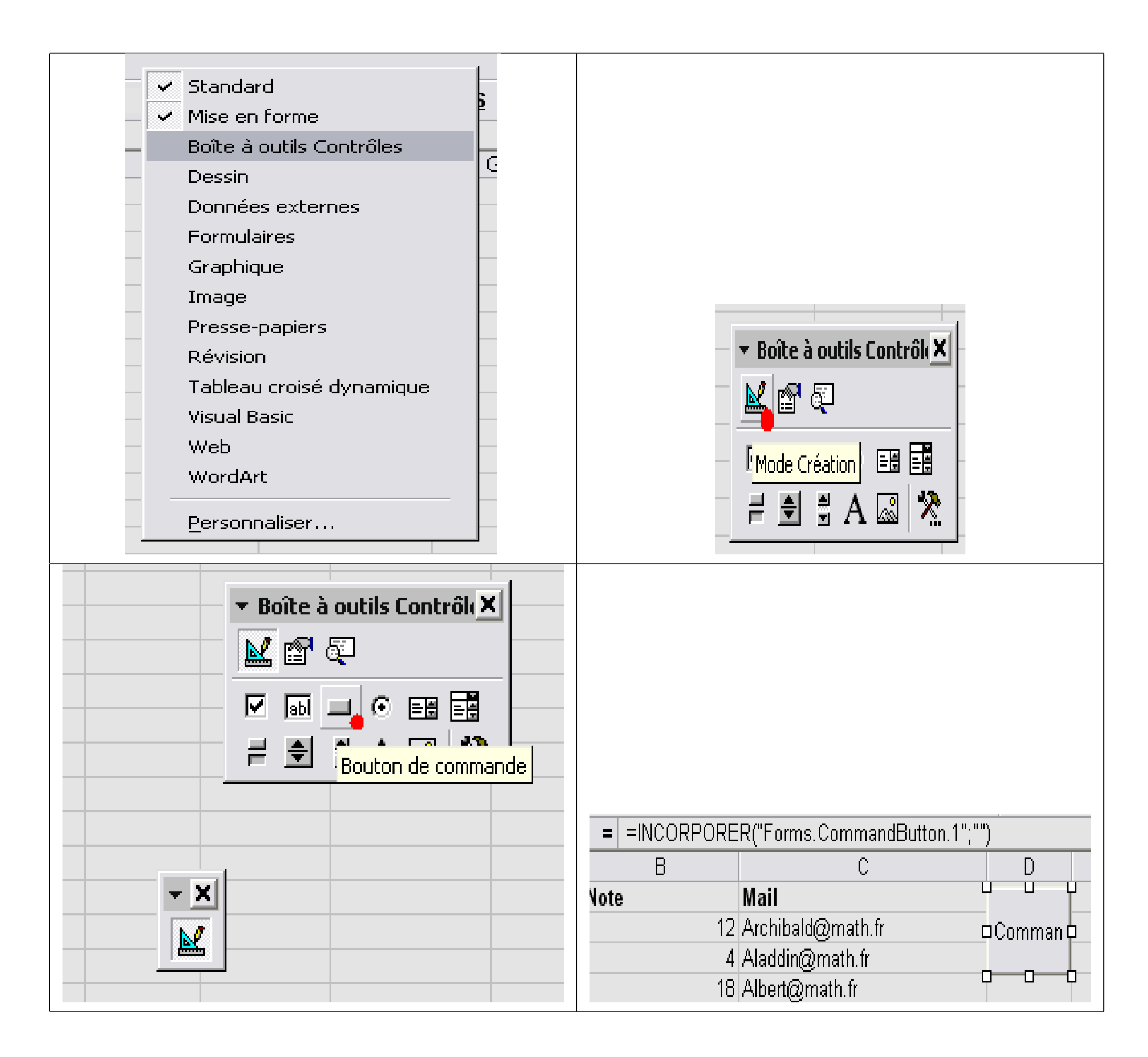


FIG. 2.7 – Bande dessinée : clic du bouton droit de la souris sur la barre d'outils, sélectionner *Boîte à Outils*. Activer le mode *Création*. Clic sur le bouton et l'insérer dans la feuille Excel.

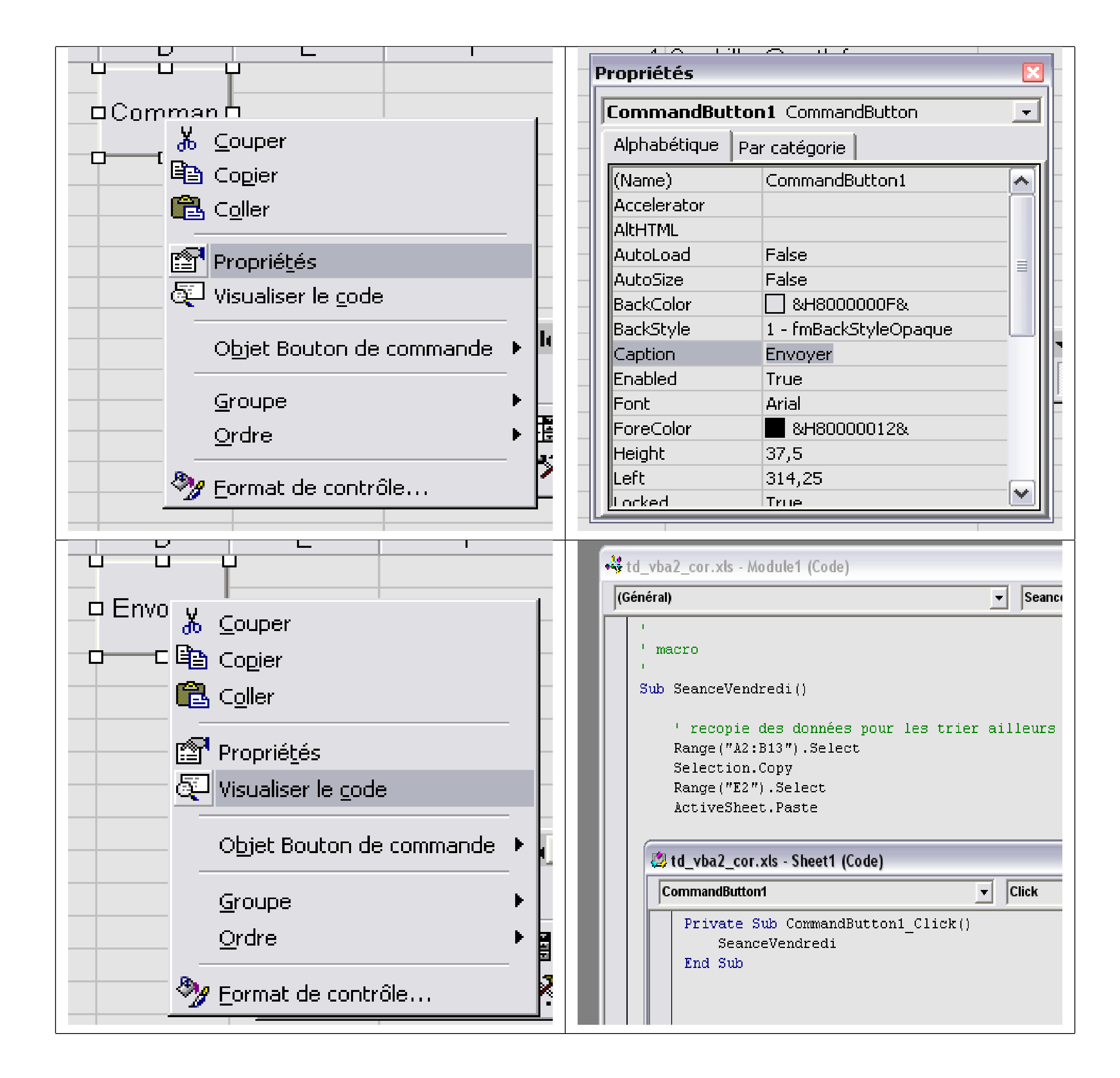


FIG. 2.8 – Bande dessinée : clic du bouton droit sur le bouton inséré, sélectionner *Propriété*. Changer l'intitulé *Caption*. Fermer la fenêtre des propriétés, choisir *Visualiser le code*. Une fenêtre s'ouvre dans l'éditeur VBA, elle contient le sous-programme qui sera exécuté à chaque pression du bouton inséré dans la feuille de calcul. C'est ici qu'il faut appeler la macro `SeanceVendredi`. Il suffit ensuite de basculer vers la feuille de calcul, de quitter le mode *Création* puis de cliquer sur le bouton de la feuille pour vérifier que tout marche bien.

9) Pour ceux qui ont encore du courage, il faudrait modifier le code de la macro pour qu'elle puisse prendre en compte un nombre pair variable d'élèves.

10) Et si le nombre d'élèves est impair, il faut former un groupe de trois.

3 TD 3 : Construction d'une présentation

(correction page 36)

Des données ont été regroupées dans une feuille Excel à partir desquelles une cinquantaine de graphes a été construite. On souhaitent présenter ces travaux sous forme d'une présentation PowerPoint qui reprenne cette multitude de graphes. Un moyen simple serait de copier un à un les cinquante graphes depuis Excel vers PowerPoint. Long et fastidieux, on propose d'utiliser VBA pour créer automatiquement la présentation depuis Excel.



Fig. 3.9 – La boîte de dialogue qui permet de lancer la construction de la présentation pour un ensemble d'année.

4 TD 4 : Construction d'une présentation à partir de photos

(correction page 42)

Après un long séjour en vacances et une grande moisson de photos numériques, un baroudeur souhaite présenter quelques extraits de ses nombreuses péripéthies. Etant tout aussi baroudeur sur un ordinateur, il opte pour une aventureuse présentation sur PowerPoint à base de programme VBA. Une heure ou deux et le tour sera joué... enfin presque.

1) La première étape consiste à récupérer la liste des images d'un répertoire contenant des images. L'aide de Visual Basic sur la fonction `Dir` vous donnera un exemple à recopier qu'il faudra adapter pour chercher des images dont les extensions sont `jpg`, `jpeg`, `tif`, `png`, `bmp`.

2) A partir de la fonction précédente, il faut maintenant créer un bouton qui exécute une procédure :

1. qui récupère le nom d'un répertoire dans la première case de la feuille de calcul,
2. qui obtient la liste de toutes les images de ce répertoire,
3. qui écrit le nom de toutes ces images dans des cases de la feuille de calcul.

Petit rappel, pour accéder ou modifier le contenu d'une case, on utilise l'instruction :

```
Cells (4,3).Value = 3  ' on modifie le contenu de la ligne 4 colonne 3
```

Il sera peut-être utile de revenir sur les séances de TD précédentes : il faut d'abord créer un bouton en mode *Design* et lui associer une procédure ou macro. A l'intérieur de cette macro, on exécute les trois tâches décrites précédemment. L'indice d'un élément d'un tableau doit apparaître en parenthèses.

3) Il faut maintenant créer une présentation PowerPoint. On ouvre l'application PowerPoint :

```
Dim power As Object
Set power = CreateObject("PowerPoint.Application")
```

On crée une présentation :

```
Dim pres As Object
Set pres = power.Presentations.Add(WithWindow:=msoTrue)
```

On rend l'application visible :

```
power.Visible = True
```

4) Ensuite, pour chaque image, il faut :

1. créer un slide, numéro `nb`
```
Set slide = pres.Slides.Add(Index:=nb, Layout:=12)
power.ActiveWindow.ViewType = 1
```
2. effectuer une petite magouille PowerPoint pour sélectionner le slide qu'on vient de créer
```
power.ActiveWindow.Selection.Unselect
power.ActiveWindow.View.GotoSlide Index:=nb
power.ActiveWindow.Selection.Unselect
```
3. insérer l'image dont le nombre est `imnom` dans le slide
```
power.ActiveWindow.Selection.SlideRange.Shapes.AddPicture _
        (Filename:=imnom, LinkToFile:=False, SaveWithDocument:=True, _
        Left:=0, Top:=0).Select
```

4. dire qu'on a terminé de manipuler le slide par l'instruction :
   `Set slide = Nothing`

5) Il suffit maintenant de répéter ces opérations pour chaque image du répertoire à l'aide d'une boucle. Quelques indices :

1. La boucle `For` ... `Next`.
2. Les plus petit et plus grand indices d'un tableau sont retournés par les fonctions `LBound` et `UBound`.
3. Si `res` est un tableau, `res(nb)` est l'élément d'indice `nb` du tableau. Dans notre cas, le tableau est celui retourné par la fonction de la question 1.

6) Les images ne sont pas toutes de tailles identiques, il faut les agrandir ou les faire rapetisser, les placer au centre. Tout d'abord une petite magouille VBA pour sélectionner l'image dont il faut modifier les propriétés :

```
power.ActiveWindow.Selection.Unselect
power.ActiveWindow.View.GotoSlide Index:=nb
power.ActiveWindow.Selection.Unselect
power.ActiveWindow.Selection.SlideRange.Shapes(1).Select
```

Grâce à ces lignes, on peut accéder ou modifier les informations suivantes :

```
With power.ActiveWindow.Selection.ShapeRange
    .Left   ' coordonnées du coin supérieur gauche
    .Top    '

    .Width  ' largeur et
    .Height ' hauteur de l'image
End With
```

La dernière information nécessaire sont les dimensions du slide. Il semblait possible de les obtenir grâce à `power.Width` et `power.Height` mais les images ne sont manifestement pas centrées. Le plus simple est, à l'intérieur d'une présentation PowerPoint, d'enregistrer une macro pendant qu'on trace une droite terminant dans le coin inférieur droit. Le code VBA de cette macro donnera les coordonnées les dimensions du slide.

Réduire, agrandir, centrer les images, tout est possible.

7) On termine la présentation en modifiant le fond (noir) et par la mise en place de transition automatique entre les slides. Le plus simple est d'enregistrer une macro qui change le fond blanc des slide en noir et qui modifie les transitions entre slides puis de recopier ce code dans le programme VBA.

8) C'est presque fini mais on voudrait que la création de la présentation soit effectuée par l'intermédiaire de la boîte de dialogue figure 4.10. Dans un premier temps, on va créer la boîte de dialogue. Il faut aller dans l'éditeur VBA et créer ce qu'on appelle un *UserForm* (figure 4.11).

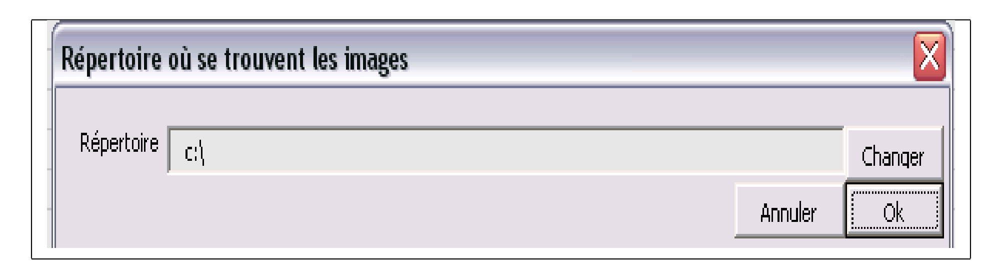


Fig. 4.10 – La boîte de dialogue qui permet de saisir le répertoire des images.

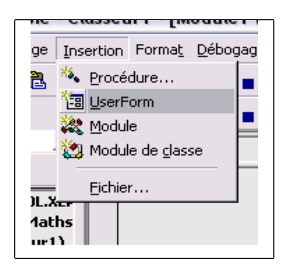


Fig. 4.11 – Comment insérer une boîte de dialogue ou UserForm.

De nombreuses fenêtres apparaissent à l'écran (figure 4.12). Tout d'abord, la boîte de dialogue elle-même, vide. A côté, une première boîte à outils qui permet d'insérer ce qu'on appelle des *contrôles*. Le terme contrôle regroupe tout ce qui permet d'interagir avec l'utilisateur. Dans notre cas, ils se résumeront à une zone de saisie permettant de récupérer un chemin, une zone de texte indiquant à quoi sert la zone de texte et trois boutons comme dans la figure 4.10. La dernière fenêtre est celle des propriétés, entre autres, il y a celle intitulée *Caption* qui est le titre de la fenêtre ou du contrôle sélectionné[4], c'est-à-dire ce qui apparaît à l'écran. Il y a aussi la propriété *Name* est le nom de variable désignant le contrôle, c'est-à-dire son nom qui le désigne dans le programme (différent de qui apparaît à l'écran).

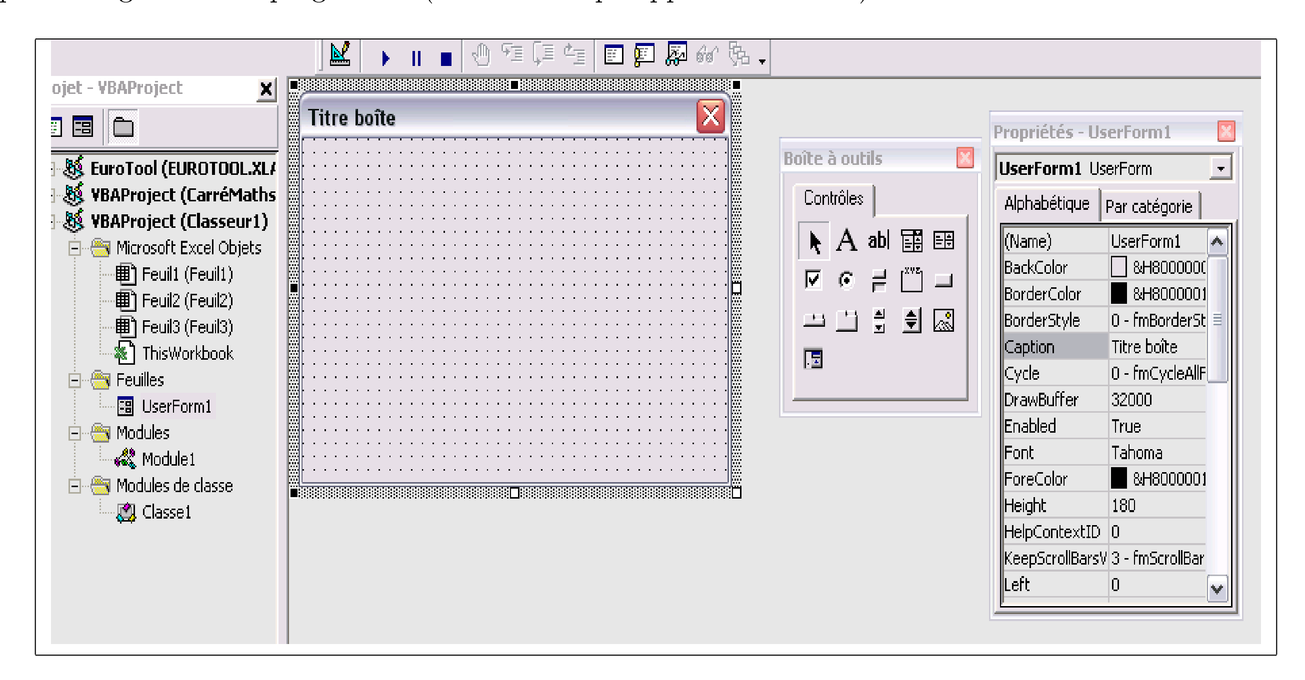


Fig. 4.12 – Trois fenêtres : celles de la boîte dialogue, celles des contrôles, celles des propriétés.

Votre première mission, si vous l'acceptez, est de réussir à imiter la boîte de dialogue de la figure 4.10. En particulier, la boîte de dialogue doit avoir comme *nom* `image_dialog` et le *nom* de la zone de saisie doit être `repertoire`.

9) Il reste à préciser ce que doit faire Excel lorsqu'on presse chacun des boutons à commencer par le premier, celui de la feuille de calcul et qui pour l'instant crée la présentation. Et au lieu de la créer, il va maintenant afficher la boîte de dialogue. On suppose que son nom (Name) est `image_dialog` (celui défini dans ses propriétés). Elle va apparaître à l'aide des trois lignes suivantes :

[4] L'object sélectionné, la boîte de dialogue, un contrôle qu'elle contient, est entourée de huit petits carrés.

```
    Load image_dialog    ' création de la boîte de dialogue
    image_dialog.repertoire.Text = Cells(1, 1).Value
       ' on stipule que dans la zone de saisie (dont le nom est repertoire),
       ' il doit y avoir lorsque la fenêtre apparaît
       ' le contenu de la première cellule de la feuille de calcul
    image_dialog.Show  ' Excel reprend le contrôle et attend vos instructions
                       ' après avoir affiché la boîte de dialogue
```

Pour l'instant, si vous cliquer sur le bouton de la feuille de calcul, la boîte de dialogue apparaît totalement inerte et sans aucune réaction face aux clics de souris excepté sur la petite croix en haut à droite.

10) On revient donc à la boîte de dialogue. Lorsqu'on clique sur le bouton *Annuler*, celle-ci doit disparaître sans rien faire. Dans l'éditeur VBA, il faut cliquer deux fois sur le bouton pour faire apparaître la procédure qui est exécutée lorsqu'on clique dessus. A cet endroit, ajoutez la ligne suivante qui détruit la boîte de dialogue :

```
' VBA ajoute automatiquement la première et la dernière de ces trois lignes
' variable_Annuler n'est pas la légende (Caption) du bouton Annuler
' mais son nom (Name)
Private Sub variable_Annuler_Click()
   Unload image_dialog
End Sub
```

11) On s'intéresse maintenant au bouton *Ok*. Il doit d'abord récupérer le répertoire de la zone de saisie et lancer la création de la présentation.

```
    Unload image_dialog    ' on détruit la boîte de dialogue
    Cells(1, 1).Value = repertoire.Text ' on récupère le répertoire saisi
                                        ' et on le met dans la première cellule de la feuille
    ' et ici : lancer la création du diaporama
```

12) Enfin, le dernier bouton qui consiste à ouvrir une autre boîte de dialogue permettant de sélectionner le répertoire :

```
Private Sub Changer_Click()
    Dim s As String
    s = BrowseForFolderShell("Choisissez un répertoire", repertoire.Text)
    repertoire.Text = s
End Sub
```

Il reste à préciser ce qu'est la fonction `BrowseForFolderShell`. Compliquée sur Excel 2000, elle est plus simple sur les versions utlérieures.

Version Excel 2000 :

```
Function BrowseForFolderShell(title As String, repertoire As String) As String
    Dim objShell As Object
    Dim objFolder As Object
    Dim strFolderFullPath As String

    Set objShell = CreateObject("Shell.Application")
```

```
    Set objFolder = objShell.BrowseForFolder(0, titre, 0, repertoire)

    If (Not objFolder Is Nothing) Then
        On Error Resume Next
        If IsError(objFolder.Items.Item.path) Then
           strFolderFullPath = CStr(objFolder): GoTo Here
        End If
        On Error GoTo 0
        If Len(objFolder.Items.Item.path) > 3 Then
            strFolderFullPath = objFolder.Items.Item.path & Application.PathSeparator
        Else
            strFolderFullPath = objFolder.Items.Item.path
        End If
    Else
        BrowseForFolderShell = repertoire
        Exit Function
    End If

Here:
    BrowseForFolderShell = strFolderFullPath

    Set objFolder = Nothing
    Set objShell = Nothing

End Function
```

Et la version Excel XP, plus simple et plus jolie (à l'écran) :

```
Function BrowseForFolderShell(title As String, repertoire As String) As String
    Dim fd As FileDialog
    Set fd = Application.FileDialog(msoFileDialogFolderPicker)
    fd.title = title
    fd.AllowMultiSelect = False
    fd.InitialFileName = repertoire & "\"

    If fd.Show = -1 Then
        Dim v As Variant
        For Each v In fd.SelectedItems
            BrowseForFolderShell = v
        Next
    Else
        BrowseForFolderShell = repertoire
    End If

    Set fd = Nothing
End Function
```

FIN.

Le baroudeur, quant à lui, a abandonné au bout d'une heure : un flamant rose est passé par là suivi par le poisson d'Arizona Dream.

En ce qui concerne la question 1, une recherche rapide sur Internet a permis de trouver l'autre moyen qui suit :

```
Function ListFileInFolder(chemin As String) As Variant
    Dim i As Long
    Dim res() As String

    ' mise en place de la liste de fichier
    With Application.FileSearch
         .NewSearch
         .FileType = msoFileTypeAllFiles
         .Filename = "*.jpg;*.jpeg;*.tif;*.png;*.bmp"
         .SearchSubFolders = False
         .LookIn = chemin
         If .Execute() > 0 Then
            ReDim res(.FoundFiles.Count)
            For i = 1 To .FoundFiles.Count
                res(i) = .FoundFiles(i)
            Next i
         End If
    End With
    ListFileInFolder = res
End Function
```

5 TD 5 : Résolution d'un Sudoku

(correction page 46)

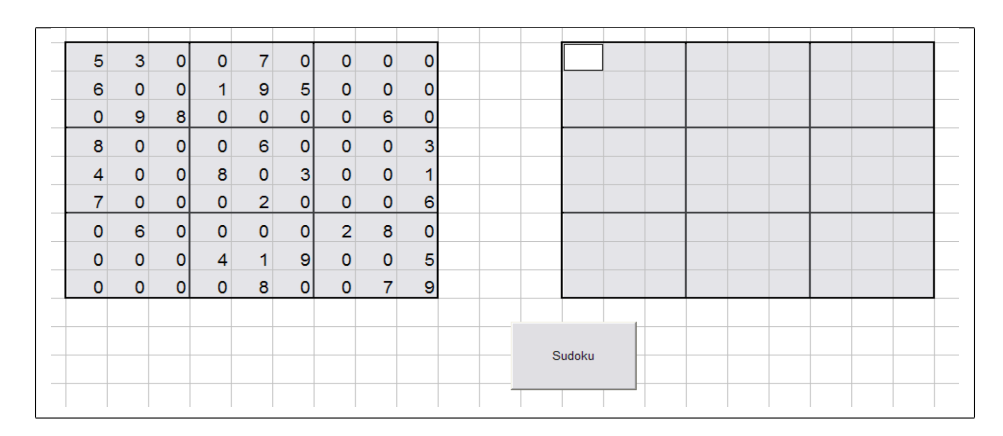
Réjoui(e) ou agacé(e) à l'idée de résoudre un Sudoku ? Et d'utiliser un ordinateur pour le résoudre à votre place ? C'est pourtant ce qui va vous arriver durant les deux heures qui suivent. La figure 5.13 montre ce qu'est un Sudoku. Ce carré incomplet qu'il s'agit de remplir obéit à certaines règles :

1. Chaque case contient un chiffre entre 1 et 9.
2. Dans chaque ligne, il ne peut y avoir deux chiffres égaux.
3. Dans chaque petit carré, il ne peut y deux chiffres égaux.

5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	2	8	0
0	0	0	4	1	9	0	0	5
0	0	0	0	8	0	0	7	9

Fig. 5.13 – Exemple de Sudoku.

A l'aide de ces informations, on va essayer de construire un programme sous Excel qui résoud les Sudoku. La figure 5.14 donne un exemple du résultat qu'on cherche à obtenir. On sélectionne la plage contenant le Sudoku, on sélectionne la plage devant recevoir le résultat, on clique sur le bouton et le tour est joué ou presque.



5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	2	8	0
0	0	0	4	1	9	0	0	5
0	0	0	0	8	0	0	7	9

Fig. 5.14 – Mode d'emploi : le Sudoku est défini par un chiffre en 1 et 9 si celui-ci est connu, 0 s'il est inconnu. On sélectionne à la souris la première plage, on appuie ensuite sur la touche *Ctrl* et on sélection encore à la souris la second plage. Il ne reste plus qu'à cliquer sur le bouton toujours à la souris.

Comment écrire ce programme ? On pourrait choisir une case vide, y poser un chiffre au hasard mais différent de ceux déjà présents sur la même ligne et dans le petit carré auquel appartient la case vide.

Une fois cette case remplie, on obtient un nouveau Sudoku qu'on cherche à nouveau à résoudre. Cette constatation nous dirige vers une fonction récursive qu'on appellera `resolution`. On peut résumer cette fonction en trois étapes :

1. on choisit une case vide
2. on place dans cette case un chiffre qui vérifie les règles d'un Sudoku
3. si toutes les cases sont pleines, c'est fini, sinon on rappelle à nouveau la fonction `resolution` sur ce nouveau Sudoku qui contient une case vide de moins.

Ce découpage suggère trois tâches bien définies. Choisir une case vide peut-être fait aléatoirement dans un premier temps. Nous verrons plus tard comment faire mieux. Pour l'instant, nous allons nous contenter d'un programme simple, qui marche même s'il est lent. La seconde étape maintenant : placer un chiffre dans une case vide, à choisir entre 1 et 9 ou mieux encore dans l'ensemble des chiffres non déjà présents sur la même ligne ou dans le petit carré. Nous allons donc écrire une fonction qui retourne l'ensemble des chiffres possibles pour une case vide. Cette fonction se présente sous cette forme :

```
Function nombre_possible_pour_case(ByRef su As Variant, _
ByVal i As Long, ByVal j As Long) As Variant
    ' ....
End Function
```

Cette fonction prend comme paramètres :

1. `su` : un tableau à deux dimensions, `su(k, l)` désigne la case ligne `k`, colonne `l`
2. `i`, `j` : deux coordonnées désignant la case pour laquelle on veut déterminer l'ensemble des chiffres possibles

Le type `Variant` désigne un type inconnu, en particulier un tableau à deux dimensions. Il existe un nom précis pour tous les types de variables simples `Long`, `String`, `Double`. Les autres comme les tableaux sont souvent désignés par le type `Variant`.

1) La première chose à faire est de vérifier que la case `i, j` est vide. La fonction `nombre_possible_pour_case` doit donc commencer par ces lignes :

```
Function nombre_possible_pour_case(ByRef su As Variant, _
ByVal i As Long, ByVal j As Long) As Variant
    Dim res() As Long     ' on crée le résultat sans connaître son contenu
    If ...........
        ReDim res(0)                  ' on crée un tableau vide
        nombre_possible_pour_case = res ' c'est le résultat de la fonction
        Exit Function                   ' on quitte la fonction
    End If
End Function
```

A vous de trouver la condition à écrire sur la troisième ligne. Si cette case n'est pas vide, il n'y a pas de chiffres possibles pour cette case et le résultat de la fonction est un tableau vide. C'est ce que font les trois lignes suivantes.

2) Si la case est vide, on doit déterminer l'ensemble des chiffres possibles pour cette case. Ou plutôt, ne serait-il pas plus facile de recenser l'ensemble des chiffres qui ne sont pas possibles, c'est-à-dire l'ensemble des chiffres déjà placés soit sur la même ligne, soit dans le même petit carré ? Il suffit pour cela de parcourir la ligne et le petit carré qui contiennent la case `i, j` et de noter tous les chiffres qu'on y rencontre. On crée donc un tableau de neuf cases qui contient `0` si le chiffre est possible et `1` si le chiffre n'est pas possible.

```
Dim paspossible(9) As Long
Dim k As Long
For k = 1 To 9
    paspossible(k) = 0  ' au départ, tous les chiffres sont possibles
Next k
```

Il faut maintenant éliminer tous les chiffres déjà présents sur la ligne `i`. On rappelle que les cases `su(k,l)` sont soit égales à `0` si elles sont vides, soit leur valeur est comprise entre `1` et `9`. C'est à vous d'écrire et de modifier le tableau `paspossible`. Quelques indices : il y a une boucle et un test.

3) On doit maintenant éliminer les chiffres présents dans le petit carré. On calcule pour cela deux entiers `ii`, `jj` grâce aux trois lignes qui suivent :

```
Dim ii, jj As Long
ii = i - ((i - 1) Mod 3)
jj = j - ((j - 1) Mod 3)
```

Que vaux `ii` lorsque `i` varie entre `1` et `9` ? Et `jj` ? Ne pourrait-on pas en déduire quelques lignes permettant d'éliminer les chiffres déjà présents dans le petit carré qui contient la case `i,j` ?

4) Maintenant qu'on a éliminé tous les chiffres pas possibles, on peut construire l'ensemble des chiffres possibles. On va d'abord les compter :

```
Dim n As Long
n = 0
For k = 1 To 9
    If ............... Then n = n + 1
Next k
```

Le résultat attendu est stocké dans la variable `n`. Il suffit juste d'écrire la bonne condition.

5) Il reste plus qu'à construire le résultat `res` de la fonction en complétant le bout de programme suivant :

```
ReDim res(n)
n = 0
For k = 1 To 9
    If ................  Then
         ...........
         ...........
    End If
Next k

' fin
nombre_possible_pour_case = res
```

Petite remarque : si on n'avait pas d'abord compté le nombre de chiffres pas possibles, on n'aurait pas pu créer le bon nombre de cases dans le tableau `res` (première ligne de l'extrait qui précède ce paragraphe).

6) La fonction `nombre_possible_pour_case` retourne donc l'ensemble des chiffres possibles pour une case vide d'un Sudoku. On va d'abord considérer que la case vide qu'on choisit de remplir est la première case vide trouvée. Il ne reste plus qu'à écrire la fonction récursive `resolution` : c'est la fonction qui résoudra le Sudoku. Elle prend comme entrée un tableau Sudoku et retourne un tableau Sudoku pour la solution. On choisit comme convention que ce tableau sera vide si la solution n'existe pas.

```
Function resolution(ByRef su As Variant) As Variant
    ' .....
End Function
```

Il y a trois étapes :

1. On trouve la première case vide.
2. On calcule l'ensemble des chiffres possibles.
3. Pour tous les chiffres possibles, on rappelle la fonction `resolution` avec une case vide en moins.

Deux questions : quand peut-on savoir que le Sudoku est résolu ? Est-il possible d'avoir un ensemble de chiffres possibles vides ? A quoi cela correspond-il et que fait-on dans ce cas-là ?

7) Ecrire le code qui permet de trouver la première case vide et écrivez ce que la fonction doit faire si cette case n'existe pas. N'hésitez pas à lire les deux questions suivantes.

8) Appelez la fonction `nombre_possible_pour_case` pour connaître l'ensemble des chiffres possibles. Que fait-on si l'ensemble est vide ? La fonction `UBound` permet de savoir si un tableau est vide ou non, regardez l'aide pour comprendre comment s'en servir.

9) La fonction `resolution` doit maintenant ressembler à quelque chose comme :

```
Function resolution(ByRef su As Variant) As Variant
    ' étape 1
    Dim i,j,vi,vj As Long
    vi = -1
    For i = 1 To 9
        For j = 1 To 9
            If su (i,j) = 0 Then
               ..................
            End If
        Next j
     Next i

     If vi = ..... Then
         resolution = ..........
         Exit Function
     End If

     ' étape 2
     Dim ens As Variant
     ens = nombre_possible_pour_case (.......)
     If UBound (ens) ...... Then
         Dim res(0) As Long
         resolution = res
         Exit Function
     End If

     ' étape 3
     Dim k As Long
     Dim copie,solution As Variant
     copie = su
     For k = 1 To .....
```

```
        copie ( ...... ) = ens (k)
        solution = resolution (copie)
        If ..... Then
            .....
            Exit Function
        End If
    Next k

    ' au fait, a-t-on réussi ou non si on arrive ici,
    ' lors de l'exécution du programme ?
    resolution = .....
End Function
```

Surtout, enregistrer votre programme avant chaque exécution : si votre programme est entré dans une boucle infini, il faudra fermer Excel de manière autoritaire (en utilisant le gestionnaire des tâches). C'est pourquoi, on propose d'ajouter une variable qui va compter le nombre d'itérations ou d'appels à la fonction `resolution`.

```
Dim nbiter As Long
Function resolution(ByRef su As Variant) As Variant
    nbiter = nbiter + 1
End Function
```

10) Est-il possible de choisir une meilleure case que la première vide ? Et si parmi toutes les cases vides, il y en avait une pour laquelle l'ensemble des chiffres possibles est vide ou ne contient qu'un seul élément ? Comment modifier la fonction `resolution` pour tenir compte de cette information et accélérer la résolution ?

11) Il ne reste plus qu'à extraire le Sudoky depuis une feuille de calcul pour l'envoyer à la fonction `resolution`, ce qui n'est pas forcément une mince affaire. Comme d'habitude, on écrit une macro qu'on pourra relier plus tard à un bouton comme lors des séances précédentes. On vérifie d'abord que l'utilisateur a bien sélectionné 81*2 cases. En VBA, l'ensemble des cases sélectionnées est `Selection`.

```
Sub macro_sudoku()
    Dim i As Long
    i = 1
    For Each ch In Selection
        i = i + 1  ' pointeur d'arrêt ici
    Next ch
    If i <> 81 * 2 Then
        MsgBox "Vous n'avez pas sélectionné 81 * 2 cases"
        Exit Sub
    End If
End Sub
```

Après avoir recopié ce petit bout de programme, placez un pointeur d'arrêt sur la cinquième ligne et regardez le contenu de `ch`. Déduisez-en le moyen d'accéder au contenu d'une case et la manière dont VBA parcourt l'ensemble des cases sélectionnées (en ligne ou en colonne ?).

12) Il faut maintenant récupérer le contenu des 81 premières cases de la sélection pour construire le tableau du Sudoku. Il y a plusieurs manières de le faire, voici l'esquisse d'une.

```
Dim sudoku(9, 9) as Long
i = 1
j = 1
For Each ch In Selection
    sudoku(i, j) = ........
    If j = 9 Then
        ......
    Else
        j = j + 1
    End If
Next ch
```

13) On lance la résolutation.

```
Dim r As Variant
nbiter = 0
r = resolution(sudoku)
```

14) Il n'y a plus qu'à remplir les 81 cases suivantes de la sélection avec la solution.

15) Deux choix s'offrent à vous dorénavant. Le premier consiste à modifier le programme pour obtenir tous les Sudoku qui vérifient la configuration de départ. Le second choix dépend de votre amour inconditionnel des Sudoku, un amour qui vous enjoint de saboter délibérément votre programme pour que jamais il n'ose encore une fois vous répondre en dix fois moins de temps qu'il ne vous en faudrait pour terminer votre Sudoku. Cela dit, si vous comptiez vous rabattre sur le démineur, il est sans doute heureux qu'il n'y ait pas de séance de VBA supplémentaire.

6 Correction du TD 1, page 2

```
' il est préférable de mettre cette ligne en haut du fichier
' afin de préciser à VBA qu'il ne doit rien faire de manière
' implicite comme utiliser une variable non déclarée
Option Explicit


'
' cette fonction prend 5 paramètres qui permettent de simuler
' l'équation stochastique de Black Scholes
'
' elle retourne un tableau de réels
'
'
Function Simulation(ByVal r As Double, ByVal sigma As Double, ByVal x0 As Double, _
            ByVal dt As Double, ByVal T As Long) As Variant


    Dim res() As Double
    Dim X As Double
    Dim i As Long
    Dim w As Double
    Dim nb As Long

    nb = T / dt + 1

    ReDim res(nb)
    X = x0
    res(0) = X

    For i = 1 To nb
        w = Rnd
        w = Application.WorksheetFunction.NormSInv(w) * dt
        X = X * (1 + r * dt + sigma * w)
        res(i) = X
    Next i

    Simulation = res

End Function



'
' définition de la macro Simulation_macro
'
Sub Simulation_macro()
```

```
Dim r As Double
Dim sigma As Double
Dim x0 As Double
Dim dt As Double
Dim T As Double

' on récupère les informations depuis la feuille Excel
r = Worksheets("Sheet1").Cells(4, 1).Value
sigma = Worksheets("Sheet1").Cells(4, 2).Value
x0 = Worksheets("Sheet1").Cells(4, 3).Value
dt = Worksheets("Sheet1").Cells(4, 4).Value
T = Worksheets("Sheet1").Cells(4, 5).Value

' on appelle la fonction simulation 5 fois
Dim i As Long
Dim marche(5) As Variant
For i = 1 To 5
    marche(i) = Simulation(r, sigma, x0, dt, T)
Next i

' on trace la courbe avec r = 0
Dim non_stochastique As Variant
non_stochastique = Simulation(r, 0, x0, dt, T)

' on récupère le nombre de points dans une solution
Dim nb As Long
nb = UBound(non_stochastique)


' on recopie les valeurs de temps et le résultats de la fonction Simulation
Dim k As Long
For i = 0 To nb
    Worksheets("Sheet1").Cells(7 + i, 1) = dt * i
    For k = 1 To 5
        Worksheets("Sheet1").Cells(7 + i, 1 + k) = marche(k)(i)
    Next k
Next i

' on recopie la solution non stochastique
k = 6
For i = 0 To nb
    Worksheets("Sheet1").Cells(7 + i, 1 + k) = non_stochastique(i)
Next i


' on met une légende
Worksheets("Sheet1").Cells(6, 1) = "temps"
Worksheets("Sheet1").Cells(6, 7) = "non stochastique"
For k = 1 To 5
```

```
    Worksheets("Sheet1").Cells(6, 1 + k) = "Simulation " & k
Next k


'
' deuxième partie
' on crée le graphique s'il n'existe pas
'

Dim nb_graphe As Long

' on compte le nombre de graphes de la feuille Sheet1
nb_graphe = Worksheets("Sheet1").ChartObjects.Count

If nb_graphe = 0 Then
    ' s'il n'y a pas alors...

    Dim plage As Range

    ' on récupère les données liées à la feuille (2 colonnes)
    Set plage = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1), _
                       Worksheets("Sheet1").Cells(7 + nb, 2))

    ' on crée un graphe
    Dim graphe_in As ChartObject
    Dim graphe As Chart
    Set graphe_in = Worksheets("Sheet1").ChartObjects.Add(100, 30, 400, 250)
    Set graphe = graphe_in.Chart

    ' on spécifie son type
    graphe.ChartType = xlXYScatterLines

    ' on lui dit quelles sont les données à dessiner,
    ' le second paramètres précise qu'elles sont organisées en colonnes
    graphe.SetSourceData plage, xlColumns

    ' on lui met un titre
    graphe.HasTitle = True
    graphe.ChartTitle.Text = "Black Scholes"

    ' on met un titre sur l'axe des Y
    graphe.Axes(xlValue, xlPrimary).HasTitle = True
    graphe.Axes(xlValue, xlPrimary).AxisTitle.Text = "Xt"

    ' on met un titre sur l'axe des X
    graphe.Axes(xlCategory, xlPrimary).HasTitle = True
    graphe.Axes(xlCategory, xlPrimary).AxisTitle.Text = "temps (jour)"

    ' on modifie le nom de la première série
    graphe.SeriesCollection(1).Name = "Courbe1"
```

```
        graphe.SeriesCollection(1).Border.Color = RGB(0, 0, 255)
        graphe.SeriesCollection(1).MarkerStyle = xlMarkerStyleNone

        ' on ajoute les séries suivantes
        Dim serie As Series
        For i = 2 To 5
            Set serie = graphe.SeriesCollection.NewSeries
            serie.XValues = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1), _
                           Worksheets("Sheet1").Cells(7 + nb, 1))
            serie.Values = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1 + i), _
                           Worksheets("Sheet1").Cells(7 + nb, 1 + i))
            serie.Name = "Courbe" & i
            serie.Border.Color = RGB(0, 0, 255)
            serie.MarkerStyle = xlMarkerStyleNone
        Next i

        ' on ajoute la solution non stochastique
        i = 6
        Set serie = graphe.SeriesCollection.NewSeries
        serie.XValues = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1), _
                           Worksheets("Sheet1").Cells(7 + nb, 1))
        serie.Values = Worksheets("Sheet1").Range(Worksheets("Sheet1").Cells(7, 1 + i), _
                           Worksheets("Sheet1").Cells(7 + nb, 1 + i))
        serie.Name = "non stochastique" & i
        serie.Border.Color = RGB(255, 0, 0)
        serie.MarkerStyle = xlMarkerStyleNone
        serie.Border.Weight = xlMedium

    End If

End Sub
```

fin correction TD 1 □

7 Correction du TD 2, page 10

```
Sub SeanceVendredi()

    ' recopie des données pour les trier ailleurs
    Range("A2:B13").Select
    Selection.Copy
    Range("E2").Select
    ActiveSheet.Paste

    ' on sélectionne la zone à trier et on trie
    Range("E2:F13").Select
    Selection.Sort Key1:=Range("F2"), Order1:=xlAscending, Header:=xlNo, _
                    OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

    ' on forme les couples
    ' le plus fort avec le moins fort
    Dim i As Long  ' position du premier
    Dim j As Long  ' position du dernier
    Dim k As Long  ' position du couple
    Dim m1 As String
    Dim m2 As String

    i = 2
    j = 13
    k = 15

    While i < j
        Cells(k, 2) = Cells(i, 5)
        Cells(k, 3) = Cells(j, 5)

        ' on fait aussi la somme des notes du groupe
        Cells(k, 4) = Cells(i, 6).Value + Cells(j, 6).Value

        k = k + 1
        i = i + 1
        j = j - 1
    Wend


    ' on crée un document Word à envoyer aux élèves

    ' on copie la sélection
    Range("B15:D20").Select
    Selection.Copy

    ' on crée un document Word
    Dim word As Object
```

```
   Set word = CreateObject("Word.Application")
   word.Documents.Add

   ' on met en forme
   word.Selection.Font.Size = 16
   word.Selection.Font.Bold = True

   ' écriture d'un petit texte dans ce nouveau document et on va deux fois à la ligne
   word.Selection.TypeText "Organisation de la classe vendredi prochain" _
                                & vbCrLf & vbCrLf

   ' on ajoute le tableau dans le document word
   word.Selection.Paste

   ' on change la mise en forme
   word.Selection.Font.Size = 12
   word.Selection.Font.Bold = False

   ' on écrit un petit message de fin
   word.Selection.TypeText vbCrLf & vbCrLf & "Bon courage et à vendredi" _
                   & vbCrLf & vbCrLf & "Votre professeur de mathématiques"

   ' sauvegarde de ce document ainsi créé :
   word.ActiveDocument.SaveAs "vendredi.doc"

   ' pour voir le résultat et éviter que Word ne se ferme tout de suite
   ' word.Visible = True
   ' MsgBox "Cliquer pour continuer"
   ' word.Visible = False

   ' on imprime le document dans un fichier posscript
   ' l'imprimante doit être installée sur votre ordinateur
   word.ActivePrinter = "HP LaserJet 5P/5MP PostScript"
   word.PrintOut Filename:="vendredi.doc", PrintToFile:=True, OutputFileName:="vendredi.ps"

   ' conversion de ps vers pdf, il faut avoir installer Miktex au préalable
   ' il est possible d'utiliser GhostScript aussi
   Shell ("C:\texmf\miktex\bin\epstopdf.exe vendredi.ps")

   ' Fermeture de ce document :
   word.ActiveDocument.Close
   Set word = Nothing

   ' envoi d'un mail
   ' voici la version sans document attaché mais elle marche sans outlook
   'ActiveWorkbook.FollowHyperlink _
Address:="mailto:xavier.dupre@wanadoo.fr?subject=cours de maths&body=groupes pour vendredi", _
NewWindow:=True

   ' on récupère les adresses dans une chaîne de caractères
```

```
    Dim tous As String
    For i = 2 To 13
        tous = tous & Cells(i, 3)
        If i < 13 Then tous = tous & ";"
    Next i


    ' envoi d'un mail avec Outlook
    Dim look As Object
    Set look = CreateObject("Outlook.Application")
    Dim email As Object
    Set email = look.CreateItem(olMailItem)
    email.to = tous
    email.Body = "organisation de la séance de vendredi, voir pièce jointe"
    email.Subject = "cours de maths"
    email.Attachments.Add "D:\Dupre\vendredi.pdf"
    email.Send

    Set email = Nothing


End Sub
```

fin correction TD 2 □

8 Correction du TD 3, page 16

La première procédure crée les graphes.

```
'
' crée deux graphes correspondant à une année
'
Sub GrapheAnnee(ByVal annee As Long)

    ' on cherche la ligne correspond à l'année
    Dim ligne As Long
    ligne = 9
    While Worksheets("table").Cells(ligne, 1) <> annee And ligne < 3000
        ligne = ligne + 1
    Wend

    If ligne = 3000 Then
        MsgBox "L'année " & annee & " est introuvable."
        Exit Sub
    End If

    ' on insère deux feuilles de calcul pour y mettre les graphes
    Dim feuille As Object
    Dim feuille_pie As Object

    On Error GoTo existe_pas

    Set feuille = Worksheets("graphes")
    Set feuille_pie = Worksheets("graphes_pie")
    GoTo suite

existe_pas:

    Set feuille = Sheets.Add
    feuille.Name = "graphes"
    Set feuille_pie = Sheets.Add
    feuille_pie.Name = "graphes_pie"

suite:

    ' '''''''''''''''''''''''''''''''''''''''''''''
    ' on crée un premier graphe
    ' '''''''''''''''''''''''''''''''''''''''''''''

    ' on récupère les données liées à la feuille (2 colonnes)
    Dim plage As Range
    Set plage = Worksheets("table").Range(Worksheets("table").Cells(ligne, 2), _
                       Worksheets("table").Cells(ligne, 4))
```

```
' création du premier graphe
Dim graphe_in As ChartObject
Dim graphe As Chart
Set graphe_in = Worksheets("graphes").ChartObjects.Add(10 + (ligne - 8) * 20, 10 + (ligne - 8)
Set graphe = graphe_in.Chart

' on spécifie son type
graphe.ChartType = xlColumnClustered

' on lui dit quelles sont les données à dessiner,
' le second paramètres précise qu'elles sont organisées en colonnes
graphe.SetSourceData plage, xlLines

' on lui met un titre
graphe.HasTitle = True
graphe.ChartTitle.Text = "Employees Year " & annee

' on assigne au graphe la légende des x
graphe.SeriesCollection(1).XValues = "=(table!R7C2,table!R7C3,table!R7C4)"
graphe.SeriesCollection(1).Name = Worksheets("table").Cells(ligne, 1)
graphe.SeriesCollection(1).Border.ColorIndex = 3
graphe.SeriesCollection(1).Interior.ColorIndex = 3

' on spécifie pas de légende
graphe.HasLegend = False

' on indique le maximum sur les ordonnées
graphe.Axes(xlValue).MaximumScale = 3500

' ''''''''''''''''''''''''''''''''''''''''''''
' on crée un second graphe
' ''''''''''''''''''''''''''''''''''''''''''''

' on récupère les données
Dim plage2 As Object
Set plage2 = Worksheets("table").Range(Worksheets("table").Cells(ligne, 5), _
                   Worksheets("table").Cells(ligne, 13))


' on crée le second graphe
Dim graphe_in_pie As ChartObject
Dim graphe_pie As Chart
Set graphe_in_pie = Worksheets("graphes_pie").ChartObjects.Add(10 + (ligne - 8) * 20, 10 + (lig
Set graphe_pie = graphe_in_pie.Chart

' on spécifie son type
graphe_pie.ChartType = xlPie

' on lui dit quelles sont les données à dessiner,
' le second paramètres précise qu'elles sont organisées en colonnes
```

```
    graphe_pie.SetSourceData Source:=plage2, PlotBy:=xlRows

    ' on lui met un titre
    graphe_pie.HasTitle = True
    graphe_pie.ChartTitle.Text = "Civilian Agencies " & annee

    ' on spécifie pas de légende
    graphe_pie.HasLegend = False

    ' on lui donne des étiquettes
    graphe_pie.SeriesCollection(1).XValues = "=table!R8C5:R8C13"

    ' on lui dit comment afficher les étiquettes
    graphe_pie.SeriesCollection(1).ApplyDataLabels Type:=xlDataLabelsShowLabel, _
        AutoText:=True, LegendKey:=True, HasLeaderLines:=True

    ' on change la couleur du fond
    graphe_pie.ChartArea.Interior.ColorIndex = 2
    graphe_pie.PlotArea.Interior.ColorIndex = 2

    ' on change quelques détails sur la position du graphe...
    graphe_pie.PlotArea.Left = 87
    graphe_pie.PlotArea.Top = 55
    graphe_pie.PlotArea.Width = 161
    graphe_pie.PlotArea.Height = 160
    graphe_pie.PlotArea.Width = 209
    graphe_pie.PlotArea.Height = 210
    graphe_pie.PlotArea.Border.LineStyle = xlNone

    ' on change la taille de la police
    With graphe_pie.SeriesCollection(1).DataLabels.Font
        .Name = "Arial"
        .FontStyle = "Normal"
        .Size = 9
    End With

    ' pas de bordure autour du graphe
    graphe_pie.ChartArea.Border.LineStyle = xlNone

End Sub
```

La seconde procédure crée la présentation.

```
'
' seconde partie : création de la présentation
'
Sub GrapheAnneeMacro_SecondePartie()

    ' on crée un document PowerPoint
    Dim power As Object
```

```
Set power = CreateObject("PowerPoint.Application")

' création d'une présentation
Dim pres As Object
Set pres = power.Presentations.Add(WithWindow:=msoTrue)

' on crée autant de pages qu'il y a de graphiques
Dim page As Object

' pour voir le résultat et éviter que Word ne se ferme tout de suite
power.Visible = True

Dim nb As Long
Dim t As Object
Dim g1 As Object
Dim g2 As Object

For nb = 1 To Worksheets("graphes").ChartObjects.Count

    ' on récupère les graphes
    Set g1 = Worksheets("graphes").ChartObjects(nb)
    Set g2 = Worksheets("graphes_pie").ChartObjects(nb)

    ' on ajoute un slide vide
    Set slide = pres.Slides.Add(Index:=nb, Layout:=12)
    power.ActiveWindow.ViewType = 1

    ' on copie le graphique comme une image
    g1.CopyPicture

    ' on l'ajoute au slide
    slide.Shapes.Paste

    ' on sélectionne l'image pour changer sa taille
    ' magouille PowerPoint
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.View.GotoSlide Index:=nb
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.Selection.SlideRange.Shapes(1).Select

    ' on modifie sa taille
    With power.ActiveWindow.Selection.ShapeRange
        .Height = 538.38
        .Width = 716.38
        .Left = 0#
        .Top = 0#
    End With

    ' on récupère le graphe pie
    ' on copie le graphique comme une image
```

```
        g2.CopyPicture

        ' on l'ajoute au slide
        slide.Shapes.Paste

        ' on sélectionne l'image pour changer sa taille
        ' magouille PowerPoint
        power.ActiveWindow.Selection.Unselect
        power.ActiveWindow.View.GotoSlide Index:=nb
        power.ActiveWindow.Selection.Unselect
        power.ActiveWindow.Selection.SlideRange.Shapes(2).Select

        ' on modifie sa taille
        With power.ActiveWindow.Selection.ShapeRange
            .Height = 250
            .Width = 350
            .Left = 350#
            .Top = 50#
        End With

        ' on fait en sorte que le fond du second graphe soit transparent
        ' de manière à ne pas masquer le premier
        power.ActiveWindow.Selection.SlideRange.Shapes("Picture 3").Select
        power.ActiveWindow.Selection.ShapeRange.Ungroup.Select
        power.ActiveWindow.Selection.SlideRange.Shapes("Picture 2").Select
        power.ActiveWindow.Selection.SlideRange.Shapes("Rectangle 4").Select
        power.ActiveWindow.Selection.ShapeRange.Fill.Visible = msoFalse
        power.ActiveWindow.Selection.SlideRange.Shapes("Rectangle 5").Select
        power.ActiveWindow.Selection.ShapeRange.Fill.Visible = msoFalse


        Set slide = Nothing

    Next nb

    ' on règle les transitions entre transparents
    With power.ActivePresentation.Slides.Range.SlideShowTransition
        .EntryEffect = 2819
        .Speed = 3
        .AdvanceOnClick = msoTrue
        .AdvanceOnTime = msoTrue
        .AdvanceTime = 1
        .SoundEffect.Type = 0
    End With


End Sub
```

Maintenant le code associé à la boîte de dialogue.

```
Private Sub annuler_Click()
    Unload graphes_dialog
End Sub

Private Sub ok_Click()

    ' on génère tous les graphes
    Dim i As Long
    Dim j As Long
    Dim a As Long

    i = Val(premier.Text)
    j = Val(dernier.Text)

    For a = i To j
        GrapheAnnee a
    Next a

    ' on créé la présentation
    GrapheAnneeMacro_SecondePartie

    ' on supprime la boîte de dialogue
    Unload graphes_dialog
End Sub
```

fin correction TD 3 □

9 Correction du TD 4, page 17

```
'
' retourne la liste des images d'un répertoire
'
Function ListFileInFolder(chemin As String) As Variant
    Dim i As Long
    Dim res() As String

    ' mise en place de la liste de fichier
    With Application.FileSearch
         .NewSearch
         .FileType = msoFileTypeAllFiles
         .Filename = "*.jpg;*.jpeg;*.tif;*.png;*.bmp"
         .SearchSubFolders = False
         .LookIn = chemin
         If .Execute() > 0 Then
            ReDim res(.FoundFiles.Count)
            For i = 1 To .FoundFiles.Count
                res(i) = .FoundFiles(i)
            Next i
         End If
    End With
    ListFileInFolder = res
End Function

'
' création d'un power point avec une liste d'images
'
Sub macro_film()

    Dim chemin As String
    Dim image As Variant
    Dim ligne As Long
    Dim colnne As Long

    ' on récupère le chemin sélectionné
    chemin = Selection.Value

    ' et sa position
    ligne = Selection.Row
    colonne = Selection.Column

    ' on récupère la liste des images
    image = ListFileInFolder(chemin)

    ' on crée un document PowerPoint
    Dim power As Object
    Set power = CreateObject("PowerPoint.Application")
```

```
' création d'une présentation
Dim pres As Object
Set pres = power.Presentations.Add(WithWindow:=msoTrue)

' on crée autant de pages qu'il y a de graphiques
Dim page As Object

' pour voir le résultat et éviter que Word ne se ferme tout de suite
power.Visible = True

' on insère les images
Dim nb As Long
For nb = 1 To UBound(image)

    ' on écrit le nom de l'image dans la cellule
    Cells(ligne + nb + 1, colonne).Value = image(nb)

    ' on ajoute un slide vide
    Set slide = pres.Slides.Add(Index:=nb, Layout:=12)
    power.ActiveWindow.ViewType = 1

    ' on sélectionne l'image pour changer sa taille
    ' magouille PowerPoint
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.View.GotoSlide Index:=nb
    power.ActiveWindow.Selection.Unselect

    ' on insère une image
    power.ActiveWindow.Selection.SlideRange.Shapes.AddPicture _
            (Filename:=image(nb), LinkToFile:=False, SaveWithDocument:=True, _
            Left:=0, Top:=0).Select

    ' on sélectionne l'image pour changer sa taille
    ' magouille PowerPoint
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.View.GotoSlide Index:=nb
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.Selection.SlideRange.Shapes(1).Select

    ' on la position en haut à gauche
    With power.ActiveWindow.Selection.ShapeRange
        .Left = 0#
        .Top = 0#
    End With

    ' on sélectionne l'image pour changer sa taille
    ' magouille PowerPoint
    power.ActiveWindow.Selection.Unselect
    power.ActiveWindow.View.GotoSlide Index:=nb
```

```
        power.ActiveWindow.Selection.Unselect
        power.ActiveWindow.Selection.SlideRange.Shapes(1).Select

        ' on modifie sa taille
        With power.ActiveWindow.Selection.ShapeRange
            ' si l'image est trop petite
            If .Height < .Width And .Width < 400 Then .Width = 400
            If .Width < .Height And .Height < 400 Then .Height = 400
            ' si l'image est trop grande
            If .Width > power.Width Then .Width = power.Width
            If .Height > power.Height Then .Height = power.Height
            ' on centre l'image
            .Left = (power.Width - .Width) / 2
            .Top = (power.Height - .Height) / 2

            ' on écrit les dimensions de l'image dans les cellules suivantes
            Cells(ligne + nb + 1, colonne + 1).Value = .Left
            Cells(ligne + nb + 1, colonne + 2).Value = .Top
            Cells(ligne + nb + 1, colonne + 3).Value = .Width
            Cells(ligne + nb + 1, colonne + 4).Value = .Height
        End With

        Set slide = Nothing
    Next nb

    ' on met un fond noir
    With power.ActivePresentation.SlideMaster.Background
        .Fill.Visible = msoTrue
        .Fill.ForeColor.SchemeColor = 2
        .Fill.Transparency = 0#
        .Fill.Solid
    End With
    With power.ActivePresentation.Slides.Range
        .FollowMasterBackground = msoTrue
        .DisplayMasterShapes = msoTrue
    End With

    ' on règle les transitions entre transparents
    With power.ActivePresentation.Slides.Range.SlideShowTransition
        .EntryEffect = 2819
        .Speed = 3
        .AdvanceOnClick = msoTrue
        .AdvanceOnTime = msoTrue
        .AdvanceTime = 1
        .SoundEffect.Type = 0
    End With
End Sub
```

Le code de la boîte de dialogue :

```
Private Sub Changer_Click()
    Dim s As String
    s = BrowseForFolderShell("Choisissez un répertoire", repertoire.Text)
    repertoire.Text = s
End Sub

Private Sub Ok_Click()
    ' on supprime la boîte de dialogue
    Unload image_dialog
    Cells(1, 1).Value = repertoire.Text
End Sub

Private Sub Annuler_Click()
    ' on supprime la boîte de dialogue
    Unload image_dialog
End Sub
```

Le module permettant de sélectionner un répertoire :

```
Function BrowseForFolderShell(title As String, repertoire As String) As String
    Dim objShell As Object
    Dim objFolder As Object
    Dim strFolderFullPath As String

    Set objShell = CreateObject("Shell.Application")
    Set objFolder = objShell.BrowseForFolder(0, titre, 0, repertoire)

    If (Not objFolder Is Nothing) Then
        On Error Resume Next
        If IsError(objFolder.Items.Item.path) Then strFolderFullPath = CStr(objFolder): GoTo Here
        On Error GoTo 0
        '// Is it the Root Dir?...if so change
        If Len(objFolder.Items.Item.path) > 3 Then
            strFolderFullPath = objFolder.Items.Item.path & Application.PathSeparator
        Else
            strFolderFullPath = objFolder.Items.Item.path
        End If
    Else
        BrowseForFolderShell = repertoire
        Exit Function
    End If

Here:
    BrowseForFolderShell = strFolderFullPath
    Set objFolder = Nothing
    Set objShell = Nothing
End Function
```

fin correction TD 4 □

10 Correction du TD 5, page 23

```
Option Explicit

Dim nbiter As Long

'
'  retourne le nombre de cases non vides
' on compte toutes celles qui ne contiennent pas 0
'
Function sudoku_cases_non_vide(ByRef su As Variant) As Long
    Dim n As Long
    Dim i As Long
    Dim j As Long

    n = 0
    For i = 1 To 9
        For j = 1 To 9
            If su(i, j) > 0 Then n = n + 1
        Next j
    Next i

    sudoku_cases_non_vide = n
End Function

'
' retourne l'ensemble des nombres possibles pour une case
' en tenant compte des contraintes
'
Function nombre_possible_pour_case(ByRef su As Variant, _
ByVal i As Long, ByVal j As Long) As Variant

    Dim res() As Long

    ' on regarde d'abord si la case est vide
    If su(i, j) > 0 Then
        ReDim res(0)
        nombre_possible_pour_case = res
        Exit Function
    End If

    ' on crée un tableau,
    ' si paspossible (i) : alors le chiffre i est déjà
    ' pris ailleurs dans la ligne, dans la colonne ou dans le petit carré
    ' qui contiennent la case i,j

    Dim paspossible(9) As Long
    Dim k As Long
```

```
For k = 1 To 9
    paspossible(k) = 0  ' au départ, tous sont possibles
Next k

' vérification des contraintes en ligne et en colonne
For k = 1 To 9
    If su(i, k) > 0 Then
        paspossible(su(i, k)) = 1
    End If

    If su(k, j) > 0 Then
        paspossible(su(k, j)) = 1
    End If
Next k


' vérification des contraintes dans le petit carré de la case i,j
Dim ii, jj, iii, jjj As Long
ii = i - ((i - 1) Mod 3)
jj = j - ((j - 1) Mod 3)

For iii = ii To ii + 2
    For jjj = jj To jj + 2
        If su(iii, jjj) > 0 Then
            paspossible(su(iii, jjj)) = 1
        End If
    Next jjj
Next iii

' nombre de possibles = tous ceux qui ne sont pas dans pospossible
' on les compte d'abord
Dim n As Long
n = 0
For k = 1 To 9
    If paspossible(k) = 0 Then n = n + 1
Next k

' puis on les met dans res
ReDim res(n)
n = 0
For k = 1 To 9
    If paspossible(k) = 0 Then
        n = n + 1
        res(n) = k
    End If
Next k

' fini
nombre_possible_pour_case = res
```

```
End Function


'
' retourne l'ensemble des nombres possibles pour une case
' en tenant compte des contraintes
'
Function get_best_solution(ByRef su As Variant) As Variant
    Dim i, j, mi, mj As Long
    Dim pos As Variant

    ' on regarde d'abord si toutes les cases sont encore viables
    For i = 1 To 9
        For j = 1 To 9
            If su(i, j) = 0 Then
                pos = nombre_possible_pour_case(su, i, j)
                If UBound(pos) = 0 Then
                    Dim r(0) As Long
                    get_best_solution = r
                    Exit Function
                End If
            End If
        Next j
    Next i

    ' on teste la case qui offre le moins de chiffres possibles vérifiant
    ' les contraintes
    Dim l As Long
    l = 0
    For i = 1 To 9
        For j = 1 To 9
            If su(i, j) = 0 Then
                pos = nombre_possible_pour_case(su, i, j)
                If UBound(pos) = 1 Then
                    Dim rrr(2) As Long
                    rrr(1) = i
                    rrr(2) = j
                    get_best_solution = rrr
                    Exit Function
                ElseIf l = 0 Or UBound(pos) < l Then
                    l = UBound(pos)
                    mi = i
                    mj = j
                End If
            End If
        Next j
    Next i

    If l > 0 Then
        ' s'il y a une solution
```

```
        Dim rr(2) As Long
        rr(1) = mi
        rr(2) = mj
        get_best_solution = rr
    Else
        ' s'il n'y en a pas
        ' excusez le nom de la variable (rrrr),
        ' la portée d'une variable en VBA est la procédure
        ' même si sa déclaration est à l'intérieur d'un bloc
        Dim rrrr(0) As Long
        get_best_solution = rrrr
    End If

End Function

'
' résolution du sudoku, su est le sudoku à résoudre
'
Function resolution(ByRef su As Variant) As Variant

    ' premier cas, le sudoku est déjà résolu,
    ' auquel cas, c'est fini
    ' la variable nbiter compte le nombre d'itération pour la résolution
    ' il vaut mieux vérifier que ce nombre ne devient pas trop grand,
    ' sinon, il est possible que le programme entre dans une boucle infinie
    ' ce qui oblige l'utilisateur à relancer Excel après l'avoir détruit l'application
    ' dans le gestionnaire des tâches
    If sudoku_cases_non_vide(su) = 81 Or nbiter > 2000 Then
        resolution = su
        Exit Function
    End If

    nbiter = nbiter + 1

    Dim copie As Variant
    copie = su

    ' retourne la case la plus sympathique
    Dim b As Variant
    b = get_best_solution(copie)

    ' s'il existe une case impossible
    If UBound(b) = 0 Then
        Dim r(0) As Variant
        resolution = r
        Exit Function
    End If

    Dim i, j As Long
    i = b(1)
```

```
    j = b(2)

    Dim nb As Variant
    Dim sol As Variant
    nb = nombre_possible_pour_case(copie, i, j)

    ' sinon on teste toutes les solutions possibles pour une case
    Dim k As Long
    For k = 1 To UBound(nb)
        copie(i, j) = nb(k)
        sol = resolution(copie)
        If UBound(sol) > 0 Then
            resolution = sol
            Exit Function
        End If
    Next k

    ' pas de solution
    Dim re(0) As Long
    resolution = re

End Function

Sub macro_sudoku()

    Dim sudoku() As Variant
    Dim i, j As Long
    Dim nb As Long
    Dim ch

    ' vérification
    i = 1
    For Each ch In Selection
        i = i + 1
    Next ch
    If i <> 81 * 2 Then
        MsgBox "Vous n'avez pas sélectionné 81 * 2 cases"
        Exit Sub
    End If

    ' on remplit le sudoku avec les 81 premières cases
    ReDim sudoku(9, 9)
    i = 1
    j = 1
    For Each ch In Selection
        sudoku(i, j) = ch.Value
        If j = 9 Then
            j = 1
            i = i + 1
            If i = 10 Then Exit For
```

```
        Else
            j = j + 1
        End If
    Next ch

    ' on résoud le sudoku
    Dim r As Variant
    nbiter = 0
    r = resolution(sudoku)

    If UBound(r) > 0 Then
        ' s'il y a une solution, on remplit les cases
        i = 1
        j = 1
        For Each ch In Selection
            If i >= 10 Then
                ch.Value = r(i - 9, j)
            End If
            If j = 9 Then
                j = 1
                i = i + 1
            Else
                j = j + 1
            End If
        Next ch
    Else
        ' s'il n'y a pas de solution, on remplit les cases de zéros
        i = 1
        j = 1
        For Each ch In Selection
            If i >= 10 Then
                ch.Value = 0
            End If
            If i = 9 Then
                i = 1
                j = j + 1
            Else
                i = i + 1
            End If
        Next ch
    End If


End Sub
```

fin correction TD 5 □